The AnimaTricks system: Animating intelligent agents from high-level goal declarations

The AnimaTricks system. Animating intelligent agents from high-level goal declarations

Vincenzo Lombardo, Fabrizio Nunnari, and Rossana Damiano

Dipartimento di Informatica and CIRMA, Università di Torino),
Virtual Reality and Multimedia Park, Torino

Abstract. This paper presents AnimaTricks, a system for the generation of the behavior of an animated agent from a high level description of its goals. The deliberation component generates a sequence of actions given a set of high-level goals. The animation component, then, translates it into an animation language, leaving to the animation engine the task of generating the actual animation.
The purpose of the system is two-fold. First, we test how deliberation can be effectively tied to the animated counterpart. Second, by generating complex animations from high-level goals, AnimaTricks supports the work of directors and animators in a pre-visualization and re–use perspective.

Key words: animation, virtual characters, multimedia production

1 Introduction

The AnimaTricks system addresses the task of realizing the animated behavior of the agents in multimedia production, with the purpose of allowing an author to specify the high–level goals of the agent, leaving the system the task of generating the actual animations that fit a specific situation.

Decision processes and actions are tightly coupled in layered architectures [9, 5, 7], a paradigm where the declarative description of the agent's behavior drives the generation of animation, in order to craft complex and non-deterministic character behavior. Following this principle, in AnimaTricks, the deliberative component is *integrated* in the system: the agent's deliberative component uses an AI planner to generate the sequence of actions that constitute the agent's behavior. The animation system, then, translates the actions into an animation language. The animation engine, based on the open source OGRE 3D rendering engine, interprets the animation language expressions to generate the visible behavior of the agent.

The structure of the paper is the following. First, we briefly survey the state of the art of declarative languages for behavior description (Sect. 2), then we describe the system architecture (Sect. 3) and the animation system (Sect. 4). Conclusions and future work end the paper.

2 Declarative languages for animated agents

The need to translate the agents' actions into animations, coping at the same time with real time constraints, has led scholars to design ad hoc animation languages that bridge the gap between behavior description, issued by some deliberative component, and the animation layer. The pioneering PAR language [2] introduced the use of templates to represent actions, establishing the design of markup languages for character animation (starting from [1]). One of the most documented and solidly implemented is the Behavior Markup Language (BML), geared to describe the multimodal communicative behavior of an agent [6]. BML acknowledges a set of communicative channels that contribute to the performance of multimodal communicative acts, such as gestures, gaze or posture. It offers tools to describe into detail the behavior of the agent along each channel (for example, the gaze is described in reference to its target, duration, etc.) and to synchronize the channels. BML descriptions can be fed to a realizer that transforms them into the corresponding animations [4]. Finally, EMBR is a animation scripting language underlying the BML realizer [4].

The family of languages mentioned above are intended to model interactive and social behavior, such as gesture, posture and so on. In Animatricks, similarly to [5], we take a neutral stance with reference to the displayed behavior, and provide an animation language geared to low–level authoring of meaningful behaviors. Apart from the differences in design goals, we identify two main differences with respect to BML/EMBR. First, it requires the specification of many temporal constraints. This can be acceptable for its use within the BML architecture, where such constraints are meant to be generated by an automatic solver, but it can be a hard job for a human editor. Differently, in Animatricks we specify animation "speeds", for which it is easier to identify default values (e.g., walk speed), and which are at run-time converted in durations according to actual path lengths. Second, EMBR lacks the syntax to expose parameters for newly defined animations. In Animatricks, the animation language supports the definition of parametrized actions which can adapt to different situations and can be stored for reuse.

3 The AnimaTricks system

The architecture of the AnimaTricks system includes three main components (Figure 1): the Planner (the mind of the "character"), the Executor (the "actor" who plays the character), and the Animation engine (the "body" of the actor).

In the current implementation of the AnimaTricks system, the Planner is given by the JSHOP2 HTN planning system [8]. According to the HTN paradigm, actions can be primitive actions, i.e., directly executable ones ("operators"), or complex actions ("methods"), if they encompass a sequence of simpler actions or abstract over alternatives. When the Planner is invoked, it matches the task to be achieved onto a high-level method and starts refining it into simpler tasks, discarding alternatives that do not fit the given world state. The

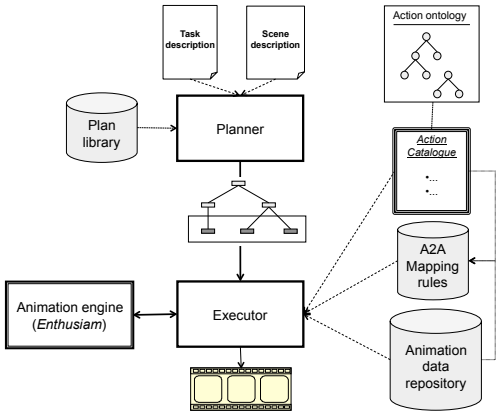

Fig. 1. The architecture of the AnimaTricks system.

refinement ends when all high-level tasks have been expanded into one or more sequences of primitive tasks.

The knowledge about plans (i.e., the plan library) is kept separated from the definition of the actions (action catalogue). By doing so, actions can be reused across different projects, independently of the plans they appear into. The actions in the catalog are annotated with reference to an ontology to guarantee that actions are externally defined in a shared, machine-readable form.

The Executor consults the action catalogue to replace each primitive task in the generated plan with the corresponding action. Then, for each action, it applies the matching A2A mapping rule to obtain the animation language expression that describes the animation.

The animation language expressions are then fed to the Animation engine, which generates the animation (possibly using pre-stored animation data from the Animation data repository). The Animation engine, implemented as part of the Enthusiasm project (http://enthusiasm.sourceforge.net), relies on the Ogre3D (www.ogre3D.org) rendering engine.

4 Creating animations: language and pipeline

The animation system interprets a list of primitive commands that generate animations edited following the techniques surveyed in [10]. The underlying primitives range from the (parametrized) playback of animation clips, to the generation of movements via IK and the blending of clips. Control structures support the parallel and sequential use of the animation primitives, and the combination of both.

Basic animations can be obtained through the *retrieval* of a clip from a repository of animation clips or through pure procedural animation, e.g., as a

bone performing an arc in the space (a *spline*). An object moves through the space following a *path*, at a certain speed, and rotates on a specified axis, at some turn speed (in degrees per second).

Finally, several animations can be blended sequentially, or, considering that each animation might operate only on a segment of the virtual agent, they be performed in parallel

For example, the following A2A (action to animation) rule (see below) says that the animation of the action of walking is generated by repeating the walking loop animation (`repeat(clipAnimation "walk_cycle")`, line 3), stored in the animation data repository, while (`par`, line 2) following the path the stretches from the initial to the final location `follow_path(from_location, to_location`, line 4). If not differently specified, the walking speed is the standard one (line 5).

```
1 walk(from_location:String, to_location:String) {
2 par(
3 {repeat(clipAnimation(‘‘walk_cycle"))
4  follow_path({from_location, to_location},
5  DEFAULT_WALK_SPEED)
6 },
7 "first")
8 }
```

The pipeline for creating the contents for the AnimaTricks system includes three main phases: the Behavior definition, where the AI expert encodes the behavior of the agent into the format required by the planner and stores it in the Plan library; the Semantic Tagging, where the basic actions are tagged with semantic labels and stored in the Action catalogue; the Action-to-Animation Mapping, where the A2A rules for actions are defined and the animation data are created.

1. **Behavior definition**. The author describes the desired behavior for a certain character and the settings in which the character may be situated. The planning expert designs and implements a plan library that encodes this behavior and tests it.
2. **Semantic tagging**. Given the rigged 3D model of the character, the animator cooperates with the knowledge engineer and the 3D programmer to describe the primitive tasks contained in the plan library, mapping them to existing actions when possible. Currently, the descriptive labels in the AnimaTricks system rely on the IEEE Standard Upper Merged Ontology (SUMO) ontology and on the WordNet lexicon [3].
3. **Action-to-animation mapping**. If an action must be produced from scratch, the animator and the 3D programmer translate it into a A2A mapping rule. First of all, the structure of the action is broken down into its components and the 3D programmer evaluates if the action can be procedurally generated. Finally, the animation data are produced and stored in the repository.

In December 2009, the AnimaTricks system was employed to conduct an experiment on the creation of extra characters in serial productions (TV series, video games, etc.). We asked an author to write down a script by inserting

actions that typically recur in TV series (for example, an interior location, some pieces of furnitures, actions such as answering the phone, walking, etc). Then, the animator animated the story following the traditional working methodology.

The same animation was produced in real time by using the procedural animation functionalities supported by Animatricks for validation purposes. The plan library included actions such as entering, sitting at the desk to accomplish several tasks, like doing or receiving phone calls, hand-writing letters and notes, getting up to take objects (pen, sheets, etc.) when necessary (Fig. 2). The plan library contained 17 complex actions (methods) and 21 primitive actions (operators). The planner was tested on 20 different scenarios and produced as many different plans, that contained from 16 to 32 actions; 5 scenarios were selected to run the evaluation.

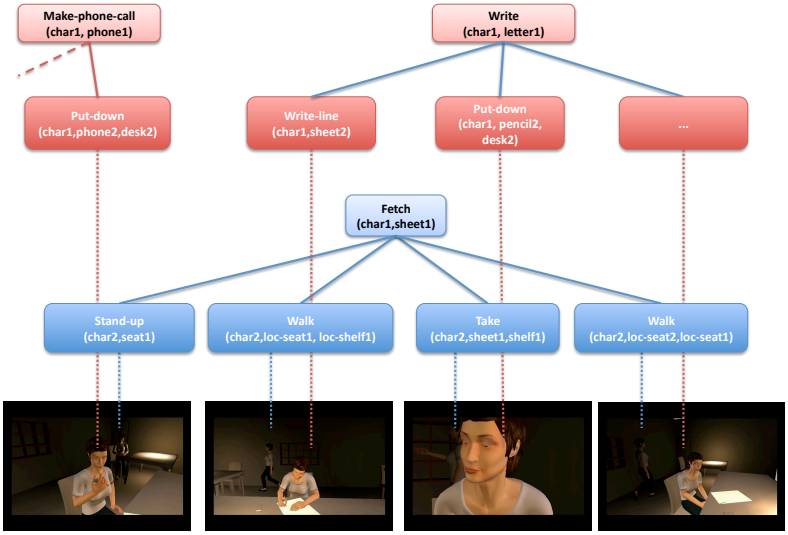

Fig. 2. Snapshots of the video generated by the AnimaTricks system. The upper part of the figure shows the plan generated by the planner (dark boxes are primitive tasks); in each shot, the characters are connected by the dashed lines to the plan actions they are executing.

The resulting video, together with the production pipeline of the AnimaTricks system, was presented to a focus group of animation producers, researchers, and trainers. The experts gave a positive evaluation to the expressiveness of the animation language and to its potential for use in the animation industry.

5 Conclusions and Future Work

In this paper we described the AnimaTricks system for animating artificial agents from a high–level specification of their behavior. The core of the system is a set of rules that map the agent's actions, generated by the AI component, to procedurally generated animations, thus tying deliberation to physical behavior in a virtual environment. With AnimaTricks, each production phase is supported by declarative languages, in order to make the behavior design more explicit and to promote the reuse of animations.

The current system does not support interactivity. Since the paradigm of HTN planning can be straightforwardly adapted to replanning, we are planning to expand the system to interactive animations.

References

1. Y. Arafa and A. Mamdani. Scripting embodied agents behaviour with CML: character markup language. In *Proceedings of the 8th international conference on Intelligent user interfaces*, page 316. ACM, 2003.
2. N. I. Badler, R. Bindiganavale, J. Allbeck, W. Schuler, L. Zhao, and M. Palmer. Parametrized action representation for virtual human agents. In J. Cassell, J. Sullivan, S. Prevost, and E. Churchill, editors, *Embodied Conversational Agents*, pages 256–284. The MIT Press, Cambridge, Massachussets, 2000.
3. C. Fellbaum et al. *WordNet: An electronic lexical database*. MIT press Cambridge, MA, 1998.
4. A. Heloir and M. Kipp. REAL-TIME ANIMATION OF INTERACTIVE AGENTS: SPECIFICATION AND REALIZATION. *Applied Artificial Intelligence*, 24(6):510–529, 2010.
5. D. Isla, R. Burke, M. Downie, and B. Blumberg. A layered brain architecture for synthetic creatures. In *INTERNATIONAL JOINT CONFERENCE ON ARTIFICIAL INTELLIGENCE*, volume 17, pages 1051–1058. Citeseer, 2001.
6. S. Kopp, B. Krenn, SC Marsella, A.N. Marshall, C. Pelachaud, H. Pirker, K.R. Thórisson, and H. Vilhjálmsson. Towards a common framework for multimodal generation: The behavior markup language. *Lecture Notes in Computer Science*, 4133:205, 2006.
7. A.B. Loyall, W. Reilly, J. Bates, and P. Weyhrauch. System for authoring highly interactive, personality-rich interactive characters. In *Proc. of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 59–68, 2004.
8. Dana Nau, Tsz-Chiu Au, Okhtay Ilghami, Ugur Kuter, Hector Munoz-Avila, J. William Murdock, Dan Wu, and Fusun Yaman. Applications of shop and shop2. *IEEE Intelligent Systems*, 20(2):34–41, 2005.
9. Ken Perlin and Athomas Goldberg. Improv: a system for scripting interactive actors in virtual worlds. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, volume SIGGRAPH '96, pages 205–216, New York, NY, USA, 1996. ACM, ACM press.
10. H. Van Welbergen, BJH Van Basten, A. Egges, Z.M. Ruttkay, and MH Overmars. Real Time Animation of Virtual Humans: A Trade-off Between Naturalness and Control. In *Computer Graphics Forum*. Wiley Online Library.